Opening the Pandora box: Generalization Theory

Prof. Filippo Fabrocini

Tongji University
School of Design & Innovation

Italy National Research Council
Institute for Computing Applications

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Quantum Neural Networks and Topological Quantum Field Theories

Antonino Marcianò[a,b], Deen Chen[c], Filippo Fabrocini[c,d,*], Chris Fields[e], Enrico Greco[f,*], Niels Gresnigt[g], Krid Jinklub[c], Matteo Lulli[h], Kostas Terzidis[c], Emanuele Zappala[i]

[a] *Center for Field Theory and Particle Physics & Department of Physics, Fudan University, Jingwan campus, Jingsan Rd, 200433 Shanghai, China*
[b] *Laboratori Nazionali di Frascati INFN, Via Enrico Fermi, 54, 00044, Frascati (Rome), Italy*
[c] *College of Design and Innovation, Tongji University, 281 Fuxin Rd, 200092 Shanghai, China*
[d] *Institute for Computing Applications "Mario Picone", Italy National Research Council, Via dei Taurini, 19, 00185, Rome, Italy*
[e] *Allen Discovery Center, Tufts University, 200 College Avenue, 02155 Medford, MA, USA*
[f] *Department of Chemical and Pharmaceutical Sciences, University of Trieste, Via Giorgieri, 1, 34127 Trieste, Italy*
[g] *Department of Physics, Xi'an Jiaotong-Liverpool University, 111 Ren'ai Rd, 215123 Suzhou, China*
[h] *Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, 1088 Xueyuan Avenue, 518055 Shenzhen, China*
[i] *Yale School of Medicine, Yale University, New Haven, 06510 CT, USA*

A R T I C L E   I N F O

Article history:
Received 18 December 2021
Received in revised form 24 February 2022
Accepted 27 May 2022
Available online 7 June 2022

Keywords:
Topological quantum field theory
Topological quantum neural networks
Graph neural networks
Quantum amplitude classifiers
Quantum perceptron

A B S T R A C T

Our work intends to show that: (1) Quantum Neural Networks (QNNs) can be mapped onto spin-networks, with the consequence that the level of analysis of their operation can be carried out on the side of Topological Quantum Field Theory (TQFT); (2) A number of Machine Learning (ML) key-concepts can be rephrased by using the terminology of TQFT. Our framework provides as well a working hypothesis for understanding the generalization behavior of DNNs, relating it to the topological features of the graph structures involved.

Control flow in active inference systems Part I: Classical and quantum formulations of active inference

Publisher: IEEE

Cite This | PDF

Chris Fields ; Filippo Fabrocini ; Karl Friston ; James F. Glazebrook ; Hananel Hazan ; Michael Levin ; ... All Authors

Abstract

Authors

Keywords

Abstract:

Living systems face both environmental complexity and limited access to free-energy resources. Survival under these conditions requires a control system that can activate, or deploy, available perception and action resources in a context specific way. In this Part I, we introduce the free-energy principle (FEP) and the idea of active inference as Bayesian prediction-error minimization, and show how the control problem arises in active inference systems. We then review classical and quantum formulations of the FEP, with the former being the classical limit of the latter. In the accompanying Part II, we show that when systems are described as executing active inference driven by the FEP, their control flow systems can always be represented as tensor networks (TNs). We show how TNs as control systems can be implemented within the general framework of quantum topological neural networks, and discuss the implications of these results for modeling biological systems at multiple scales.

Deep Neural Networks as the Semi-classical Limit of Topological Quantum Neural Networks: The problem of generalisation

Antonino Marcianò,[1] Deen Chen,[2] Filippo Fabrocini,[3] Chris Fields,[4] Matteo Lulli,[5] and Emanuele Zappala[6]

[1] *Center for Field Theory and Particle Physics & Department of Physics, Fudan University, Jingwan campus, Jingsan Rd, 200433 Shanghai, China and Laboratori Nazionali di Frascati INFN, Via Enrico Fermi, 54, 00044 Frascati (Rome), Italy, EU and INFN sezione Roma "Tor Vergata", 00133 Rome, Italy, EU*[*]

[2] *College of Design and Innovation, Tongji University, 281 Fuxin Rd, 200092 Shanghai, China*[†]

[3] *College of Design and Innovation, Tongji University, 281 Fuxin Rd, 200092 Shanghai, China and Institute for Computing Applications "Mario Picone", Italy National Research Council, Via dei Taurini, 19, 00185 Rome, Italy, EU*[‡]

[4] *Allen Discovery Center, Tufts University, 200 College Avenue, 02155 Medford, MA, US*[§]

[5] *Department of Mechanics and Aerospace Engineering, Southern University of Science and Technology, 1088 Xueyuan Avenue, 518055 Shenzhen, China*[¶]

[6] *Yale School of Medicine, Yale University, 300 George Street, New Haven, CT, 06511, USA*[**]

Deep Neural Networks miss a principled model of their operation. A novel framework for supervised learning based on Topological Quantum Field Theory that looks particularly well suited for implementation on quantum processors has been recently explored. We propose the use of this framework for understanding the problem of generalization in Deep Neural Networks. More specifically, in this approach Deep Neural Networks are viewed as the semi-classical limit of Topological Quantum Neural Networks. A framework of this kind explains easily the overfitting behavior of Deep Neural Networks during the training step and the corresponding generalization capabilities.

I. INTRODUCTION

Deep neural networks (DNNs), i.e. neural networks with several hidden layers, have become popular due to their success in a variety of learning task ranging from

the operational behavior of a non-linear weighted model trained over hundreds of thousands of inputs that contribute microscopically to the final configuration of the network. These issues together constitute the main technical challenge for achieving a fair, accountable, and transparent Artificial Intelligence (AI), with the first

UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

Chiyuan Zhang[*]
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht[†]
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

Understanding quantum machine learning also requires rethinking generalization

Elies Gil-Fuster,[1,2] Jens Eisert,[1,2,3] and Carlos Bravo-Prieto[1]

[1]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[2]*Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany*
[3]*Helmholtz-Zentrum Berlin für Materialien und Energie, 14109 Berlin, Germany*

Quantum machine learning models have shown successful generalization performance even when trained with few data. In this work, through systematic randomization experiments, we show that traditional approaches to understanding generalization fail to explain the behavior of such quantum models. Our experiments reveal that state-of-the-art quantum neural networks accurately fit random states and random labeling of training data. This ability to memorize random data defies current notions of small generalization error, problematizing approaches that build on complexity measures such as the VC dimension, the Rademacher complexity, and all their uniform relatives. We complement our empirical results with a theoretical construction showing that quantum neural networks can fit arbitrary labels to quantum states, hinting at their memorization ability. Our results do not preclude the possibility of good generalization with few training data but rather rule out any possible guarantees based only on the properties of the model family. These findings expose a fundamental challenge in the conventional understanding of generalization in quantum machine learning and highlight the need for a paradigm shift in the design of quantum models for machine learning tasks.

I. INTRODUCTION

Quantum devices promise applications in solving computational problems beyond the capabilities of classical computers [1–5]. Given the paramount importance of machine learning in a wide variety of algorithmic applications that make predictions based on training data, it is a natural thought to investigate to what extent quantum computers may assist in tackling machine learning tasks. Indeed, such tasks are commonly listed among the most promising candidate applications for near-term quantum devices [6–9]. To date, within this emergent field of *quantum machine learning* (QML) a body of literature is available that heuristically explores the potential of improving learning algorithms by having access

alization is unable to explain the great success of large-scale deep convolutional neural networks. These networks, which display orders of magnitude more trainable parameters than the dimensions of the images they process, defied conventional wisdom concerning generalization.

Employing clever randomization tests derived from non-parametric statistics [60], the authors of Ref. [59] exposed cracks in the foundations of Vapnik's theory and its successors [61], at least when applied to specific, state-of-the-art, large networks. Established complexity measures, such as the well-known VC dimension or Rademacher complexity [62], among others, were inadequate in explaining the generalization behavior of large classical neural networks. Their findings, in the form of numerical experiments, directly challenge

What is Artificial Intelligence: A definition

Today most of the main textbooks* define AI as "the study and design of intelligent agents, in which an intelligent agent is a system that perceives its environment and takes actions that **maximize** its chances of success".

- In this approach being rational means **maximizing the expected utility**.
- Yet rationality only concerns what decisions are made (**not** the thought processes behind them)

(*cf., for instance, Norvig & Russell)

Machine Learning as Function Approximation

Problem Setting:

- Set of possible instances X
- Set of possible labels Y
- Unknown target function $f : X \rightarrow Y$
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$

Input:

superscript: ith training example

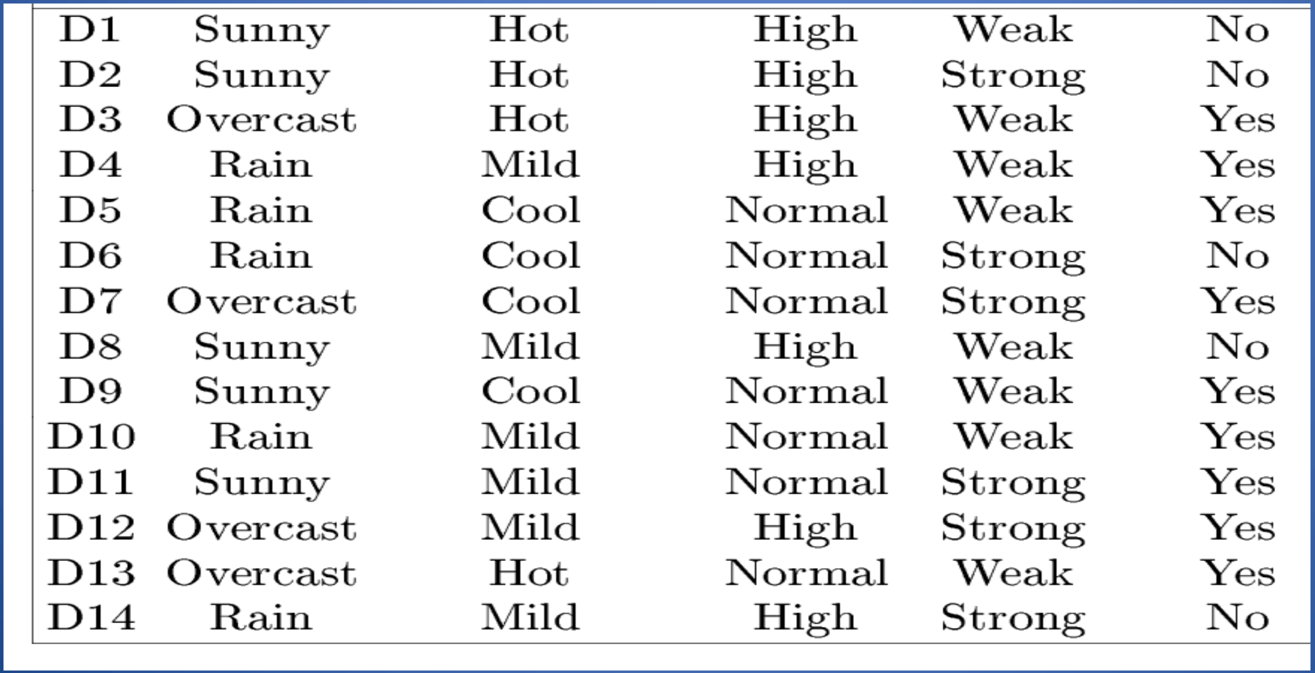
- Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function *f*

Output:

- Hypothesis $h \in H$ that best approximates target function *f*

A Training Data Set

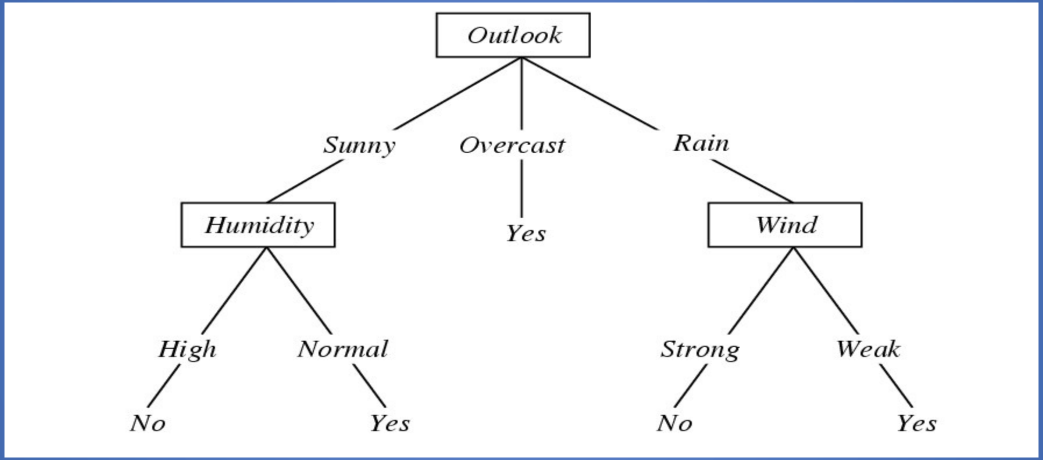
Simple Training Data Set



Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

A Decision Tree

A Decision Tree for
f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?



Each internal node: test one discrete-valued attribute X_i

Each branch from a node: selects one value for X_i

Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

Information Gain

Entropy

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

Specific conditional entropy $H(X|Y=v)$ of X given $Y=v$:

$$H(X|Y=v) = -\sum_{i=1}^{n} P(X=i|Y=v)\log_2 P(X=i|Y=v)$$

Conditional entropy $H(X|Y)$ of X given Y :

$$H(X|Y) = \sum_{v \in values(Y)} P(Y=v)H(X|Y=v)$$

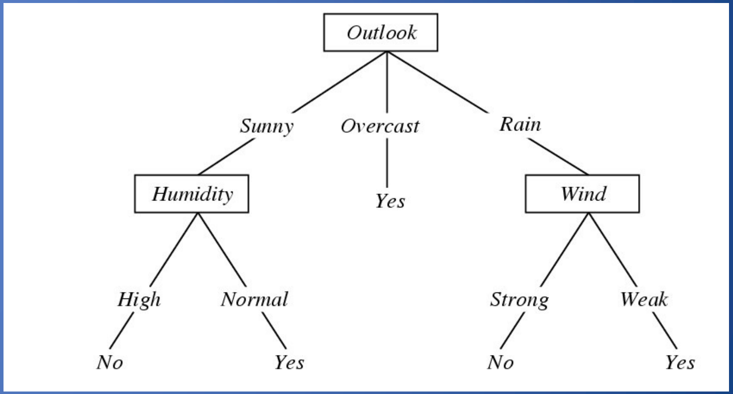
Mutual information (aka Information Gain) of X and Y :

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Decision Tree Learning

Problem Setting:

- Set of possible instances *X*
  - each instance *x* in *X* is a feature vector
  - e.g., <*Humidity=low, Wind=weak, Outlook=rain, Temp=hot*>
- Unknown target function *f* : *X* → *Y*
- *Y*=1 if we play tennis, else 0
- Set of function hypotheses *H*={ *h* | *h* : *X* → *Y* }
  - each hypothesis *h* is a decision tree
  - sorting *x* to a leaf, which assigns *y*



Input:

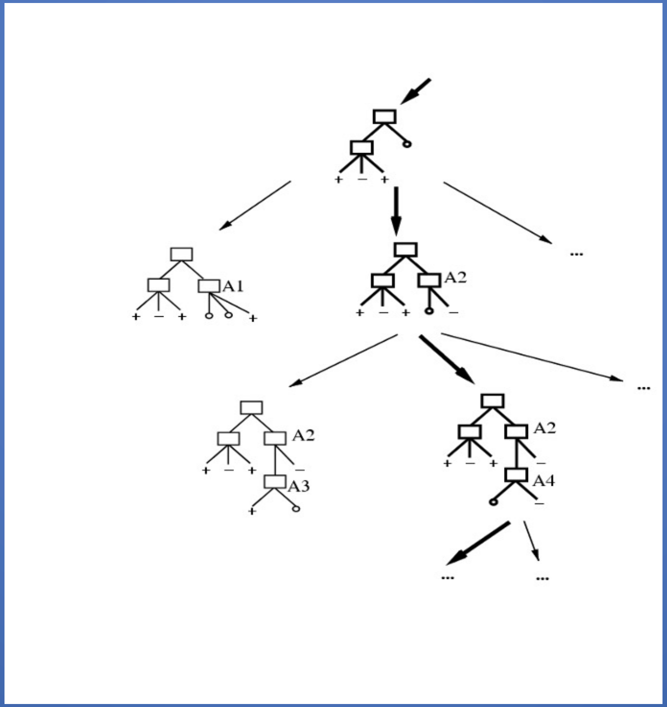
- Training examples {<$x^{(i)}$,$y^{(i)}$>} of unknown target function f

Output:

- Hypothesis $h \in H$ that best approximates target function f

Machine Learning: A first view point

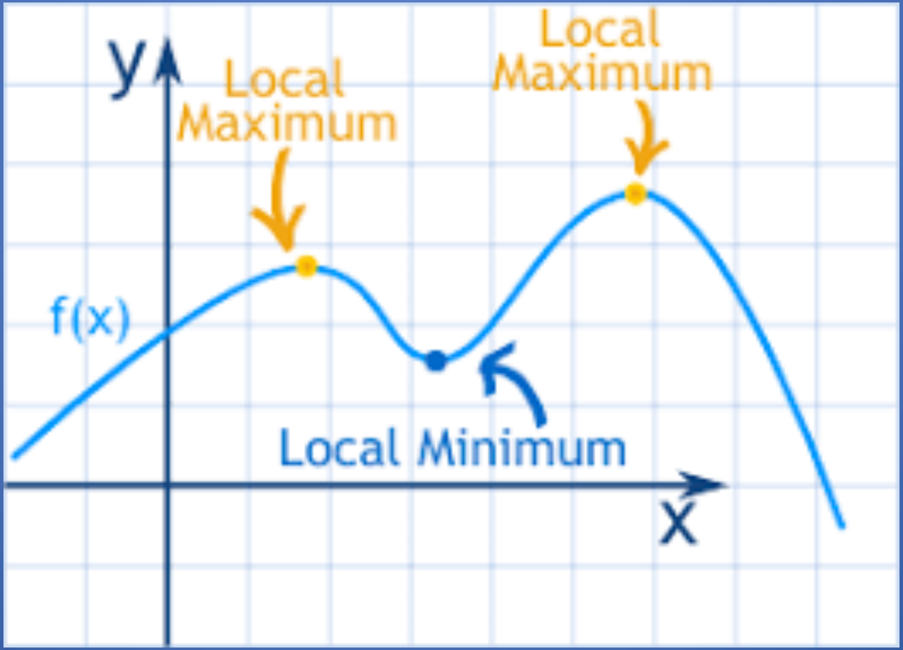
Machine learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation. The goal of this search is to find the hypothesis that best fits the training examples.



It is important to note that by selecting a hypothesis space, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever learn (*algorithmic bias*).

Machine Learning: A second viewpoint

Machine learning can be also viewed as the task of optimizing a function by avoiding the local minima. The goal of this search is to find the function that best approximates the target function.

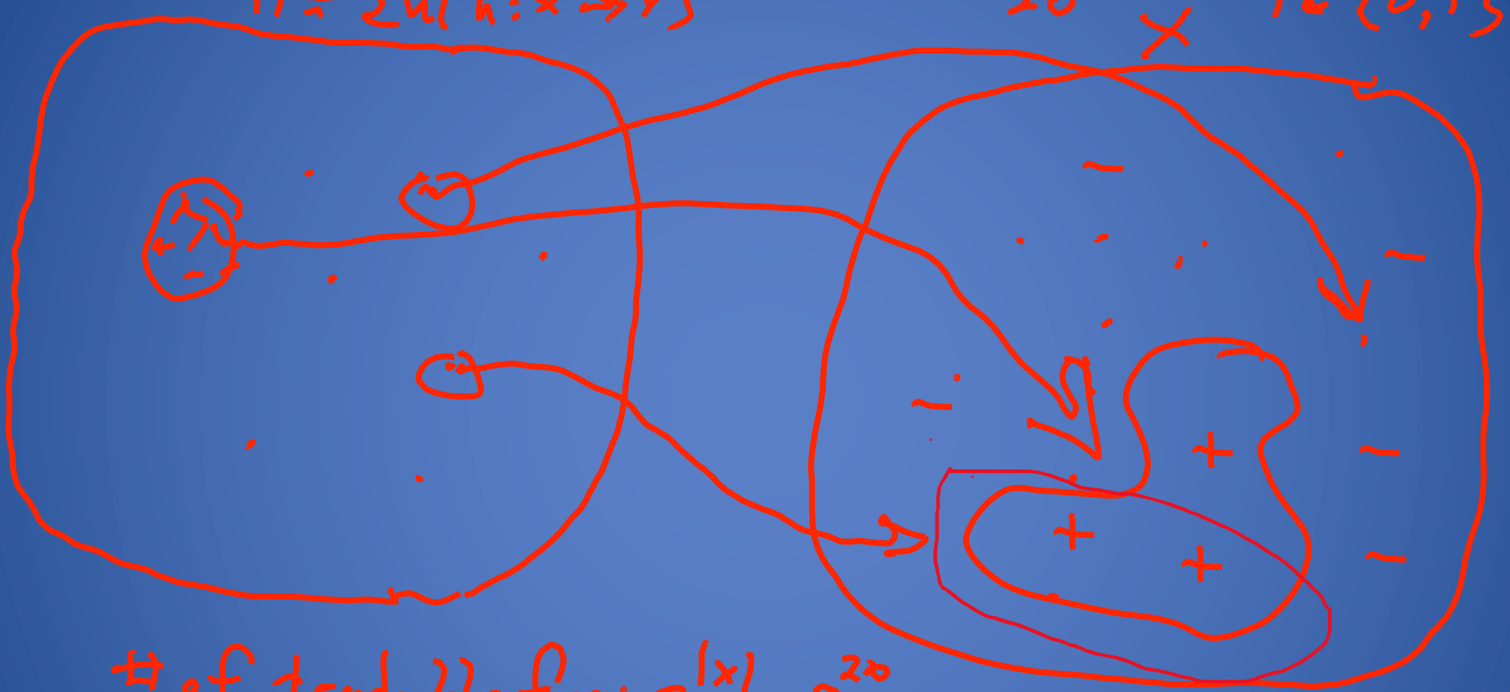


The drawback of this picture is that it ignores the role of generalization along the process.

Function Approximation: The Big Picture

$f: X \to Y$ $\quad X = \langle x_1 \cdots x_n \rangle$ $\quad x_i \in \{0,1\}$

$H = \{h \mid h: X \to Y\}$ $\quad n \leftarrow 20$ $\quad y \in \{0,1\}$



$|X| = 2^n \quad 2^{20}$

# of trainable fns: $2^{|X|} = 2^{2^{20}}$

# of fns DTs can repr $= 2^{2^{20}}$

No free lunch!

A short dictionary

Training set. It is the set of labelled samples used to train the model.

Test set. It is the set of labelled samples used to test the model.

Accuracy. It is the performance of the model on the test set corresponding to the total number of correct predictions over the total number of predictions.

Generalization. It is the ability of the learner to find a hypothesis that is able to enlarge successfully its own predictions from the training samples to the test samples.

Bias. We bias the algorithm towards a particular set of hypotheses by restricting the learner to choosing the predictor h from a specific class of functions H.

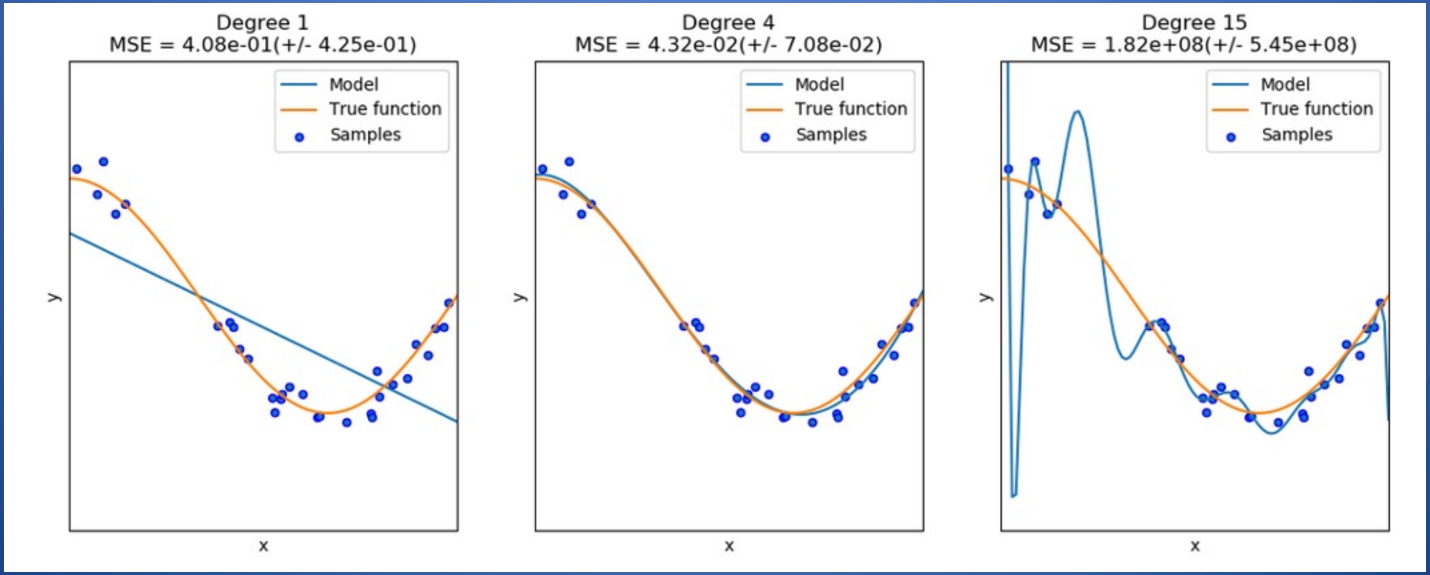
Capacity. It is defined as the ability of the model to fit a wide variety of functions: specifically, the capacity of a model specifies the class of functions H (the hypothesis class) from which the learning algorithm can choose the specific function h.

Underfitting/Overfitting (1)

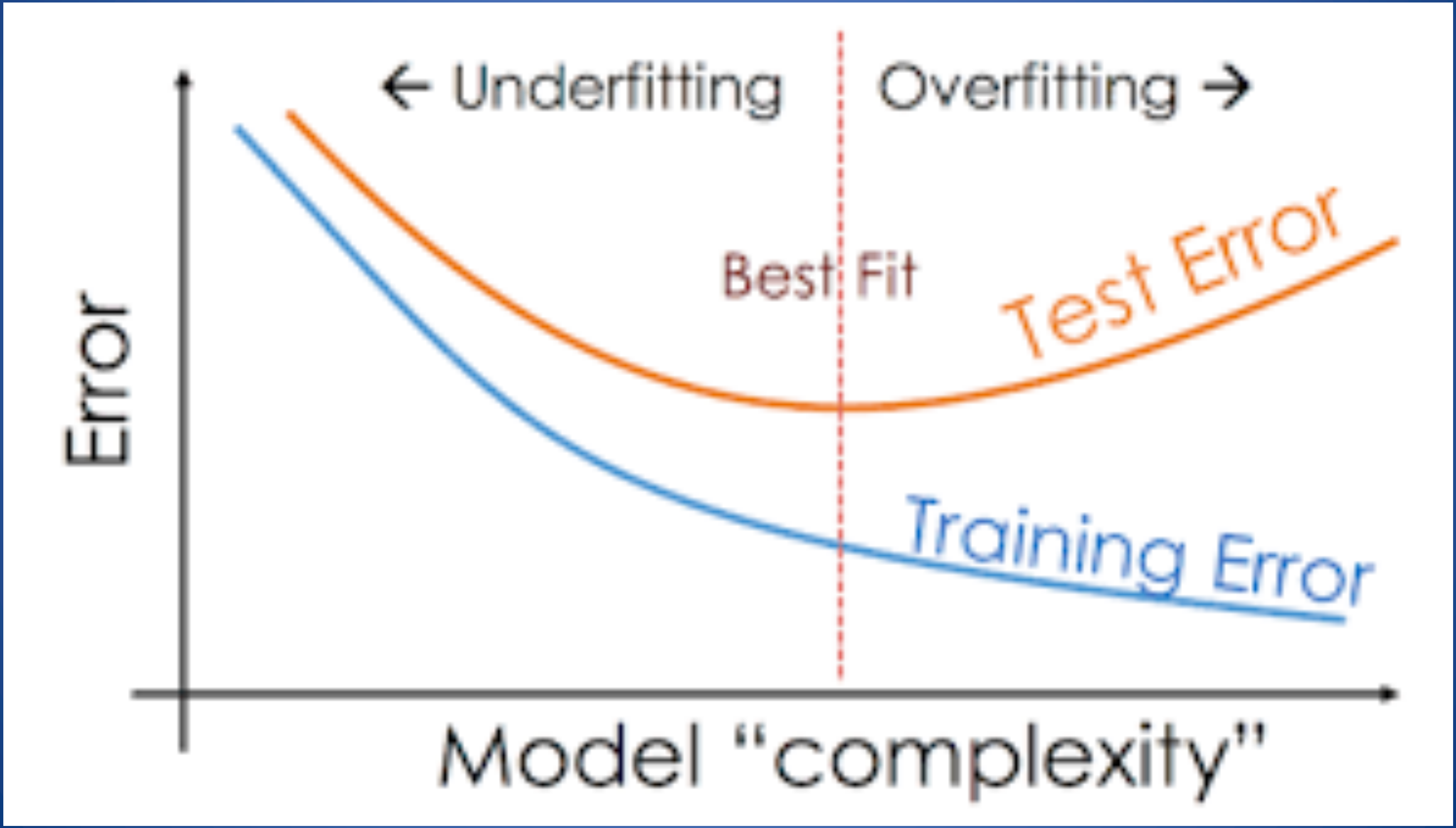
A model is *underfitting* when it is not able to achieve a sufficiently low error on the training set.

A model is *overfitting* when the gap between training error and test error is too large.

The goal of the learner is to find the right trade-off between the underfitting/overfitting regimes.



Underfitting/Overfitting (2)



Overfitting

Consider a hypothesis h and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

We say h overfits the training data if

$$error_{true}(h) > error_{train}(h)$$

Amount of overfitting = $error_{true}(h)$ - $error_{train}(h)$

Generalization, bias & capacity

Let us have a hypothesis space generated by a linear regression algorithm. It will include the set of all linear functions of its input.

If we decide to move up the level of abstraction of a linear regression algorithm to include polynomials in its hypothesis space, we will increase the capacity of the model. A polynomial of degree 1 gives us the linear regression model, with prediction:

$$y = w * x + b$$

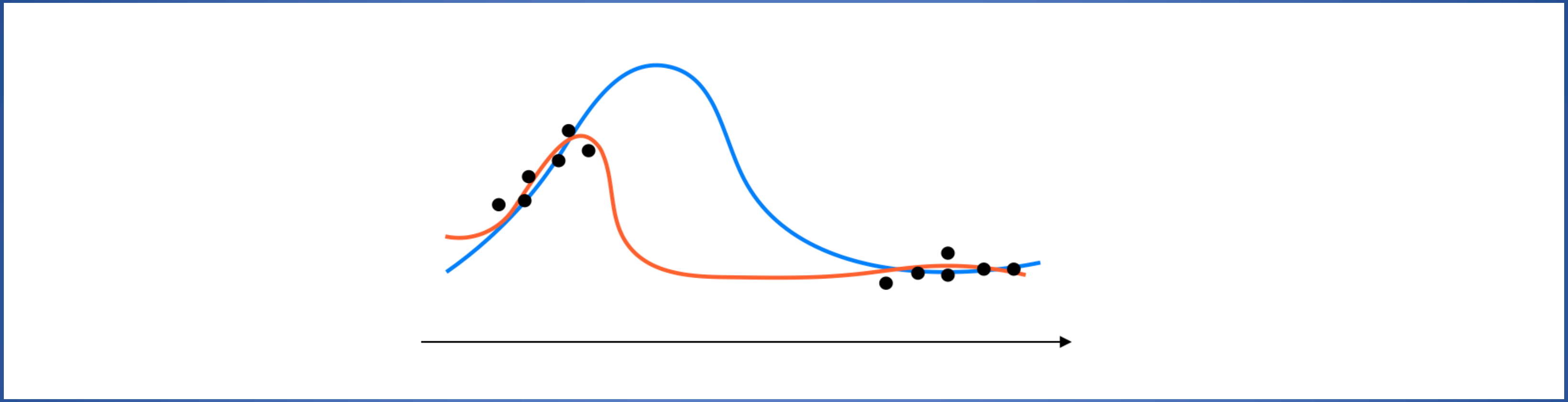
If we introduce x^2as another feature provided to the linear regression model, we can learn a model that is quadratic:

$$y = w_2 * x^2 + b$$

We could increase further the capacity of the model till to obtain a polynomial of degree 15:

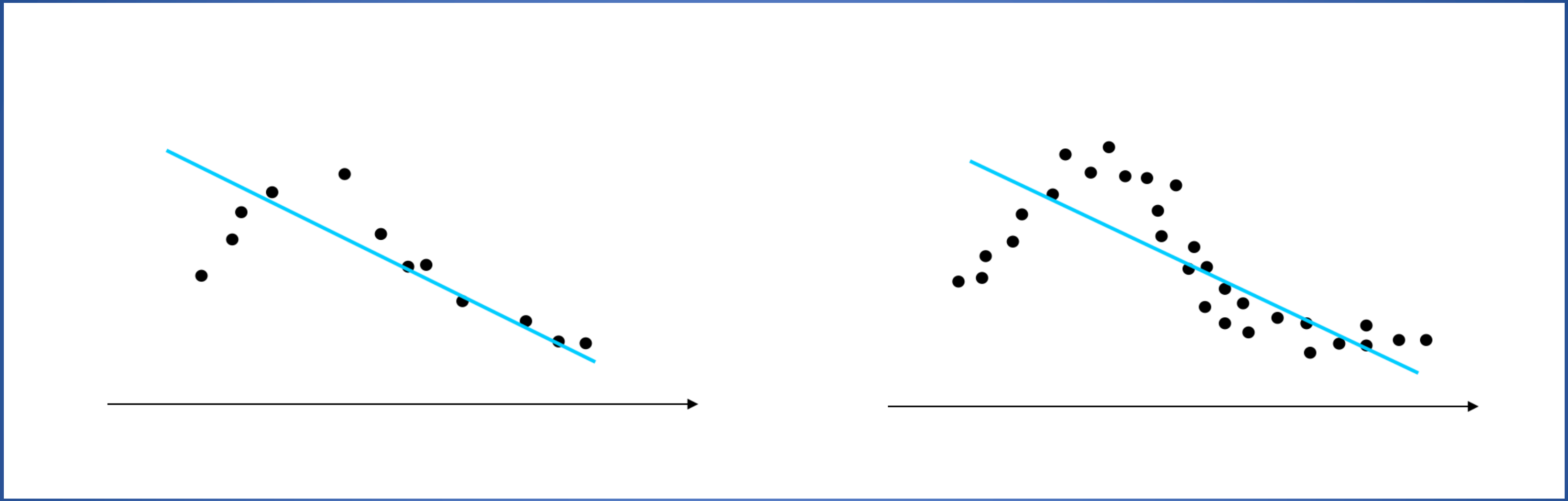
$$y = \sum_{i=1}^{15} w_i x^i + b$$

Increasing the volume of data: Case (1)



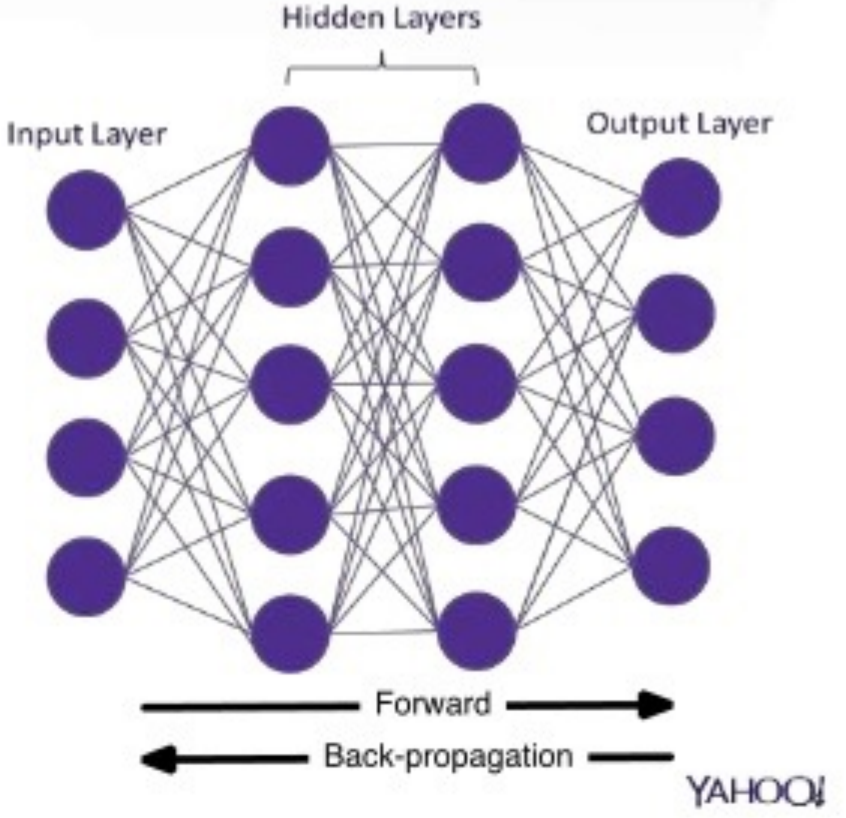
The additional data is provided where the machine learning model needs it most. In the example above, we are not going to see much improvement if we keep adding data to the tails while ignoring the middle area.

Increasing the volume of data: Case (2)



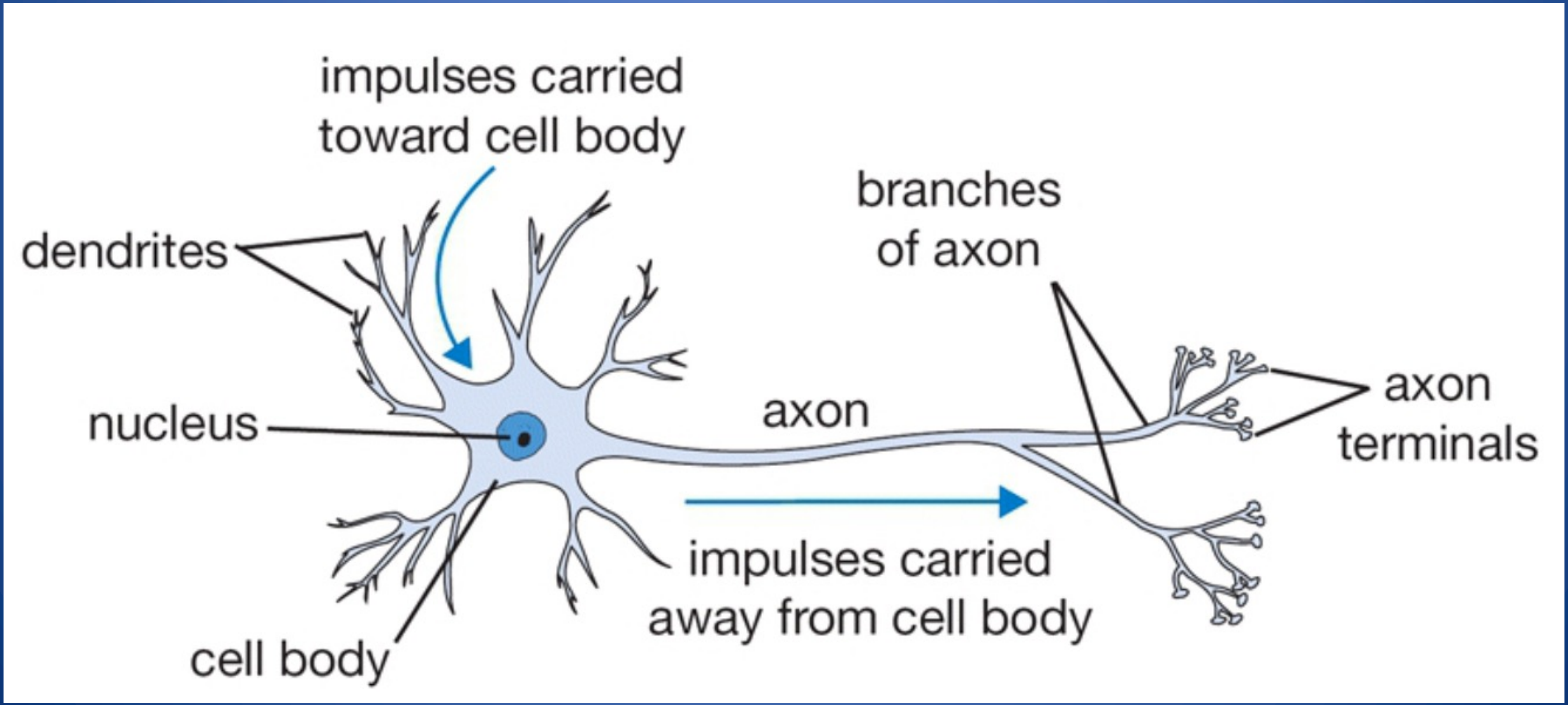
In this example, adding more data will not improve accuracy since we are attempting to model a nonlinear phenomenon with the wrong tool - a linear model.

Deep Learning

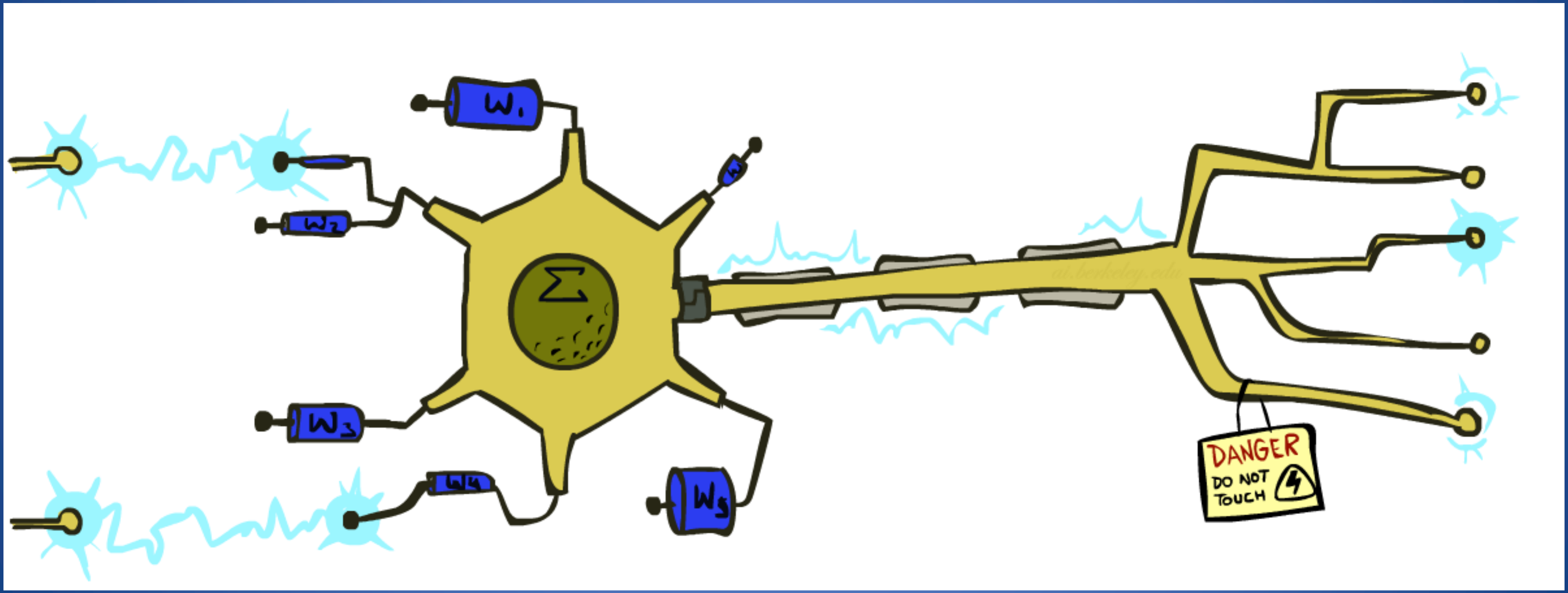


Some (simplified) biology

- Very loose inspiration: human neurons


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

Perceptron



Error-Driven Classification

Errors, and What to Do

- Examples of errors

```
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the
latest version of OmniPage Pro, for just $99.99* - the
regular list price is $499! The most common question we've
received about this offer is - Is this genuine? We would like
to assure you that this offer is authorized by ScanSoft, is
genuine and valid. You can get the . . .
```

```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are
there. We hope you enjoyed receiving this message. However,
if you'd rather not receive future e-mails announcing new
store launches, please click . . .
```

What to Do About Errors

- Problem: there is still spam in your inbox

- Need more **features** – words aren't enough!
  - Have you emailed the sender before?
  - Have 1M other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

Classifiers

X　　*f(X)*　　*y*

```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

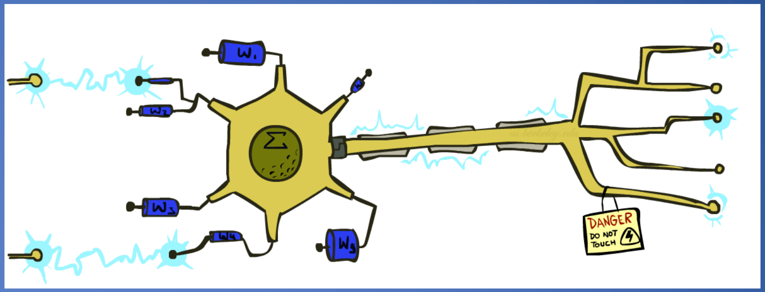
SPAM
or
+

```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

"2"

Linear Classifiers

- Inputs are **feature values**
- Each feature has a **weight**
- Sum is the **activation**



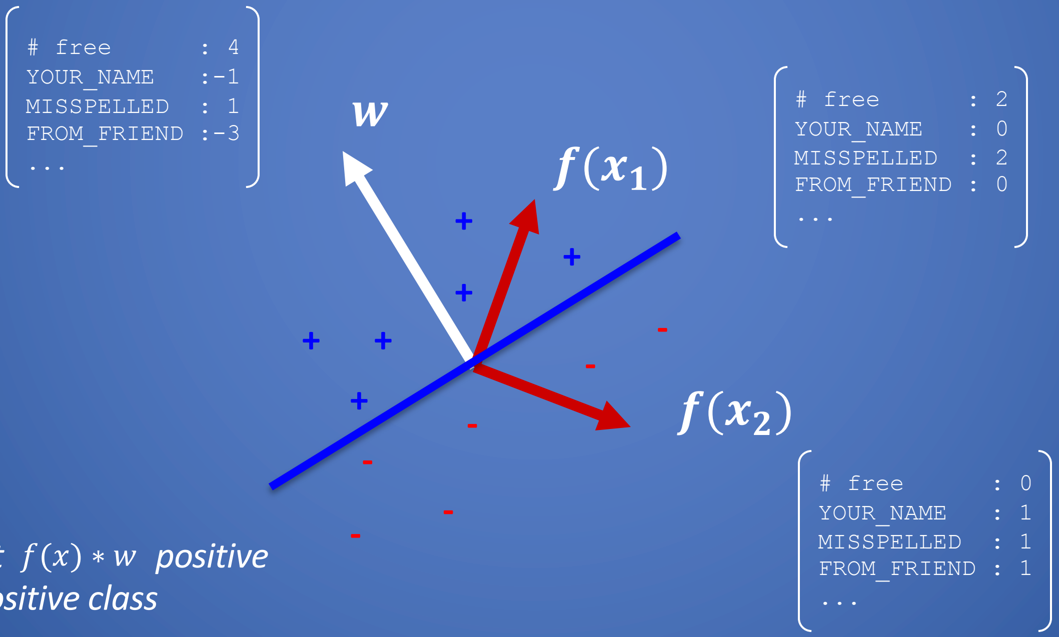
$$activation_w(x) = \sum_i w * x_i$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1

$$\begin{tikzcd}
f_1 \arrow[dr, "w_1"] & & & \\
f_2 \arrow[r, "w_2"] & \Sigma \arrow[r] & > 0? \arrow[r] & {} \\
f_3 \arrow[ur, "w_3"'] & & &
\end{tikzcd}$$

Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples



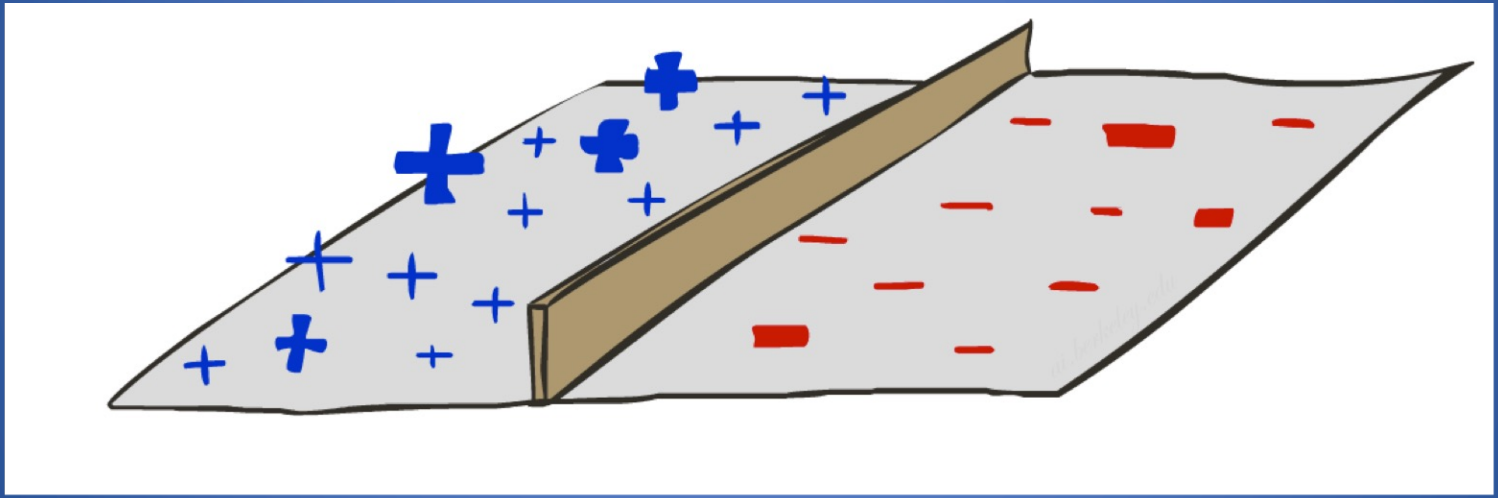
```
# free       : 4
YOUR_NAME    :-1
MISSPELLED   : 1
FROM_FRIEND  :-3
...
```

```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```

```
# free       : 0
YOUR_NAME    : 1
MISSPELLED   : 1
FROM_FRIEND  : 1
...
```

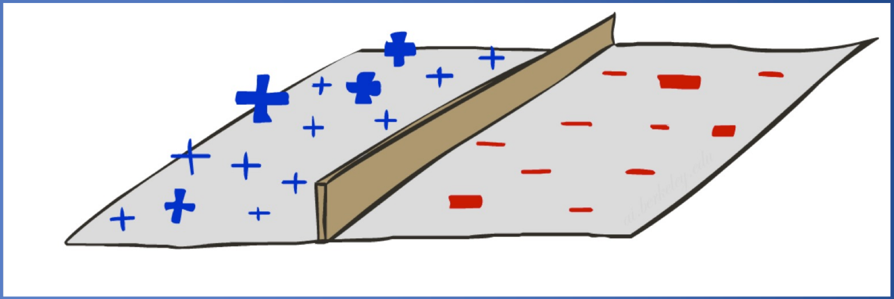
Inner product $f(x) * w$ *positive means the positive class*

Decision Rules



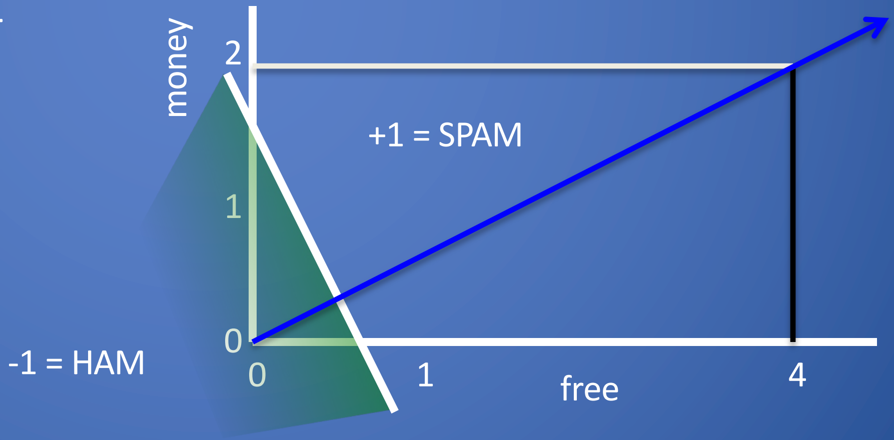
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y= +1
  - Other corresponds to Y= -1



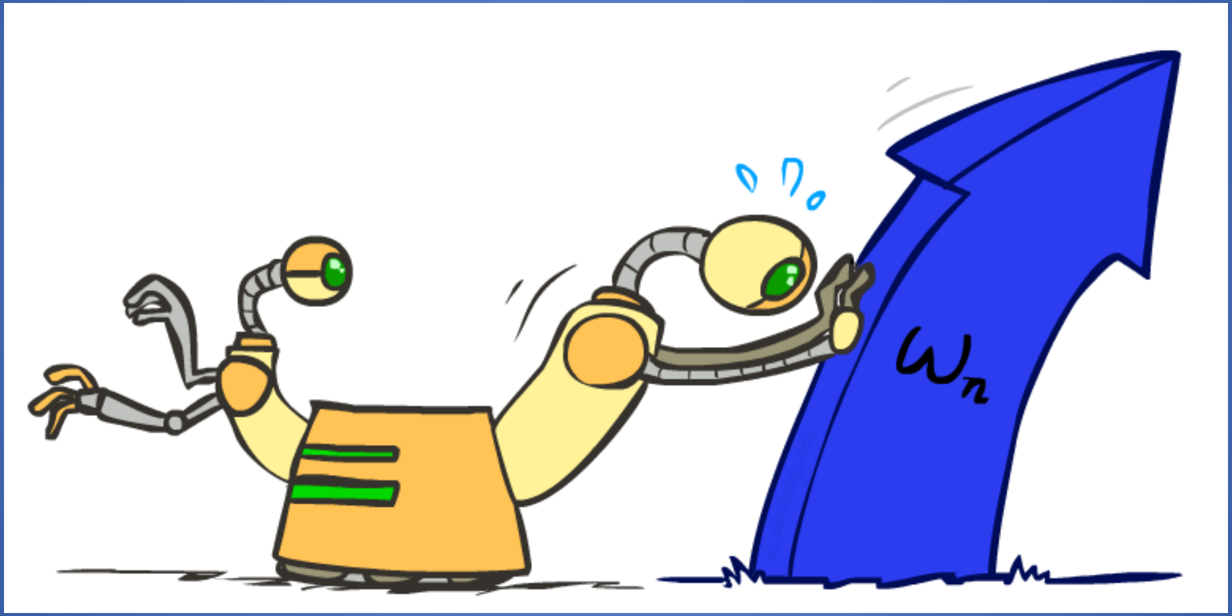
$\boldsymbol{x}$

```
BIAS  : -3
free  :  4
money :  2
...
```



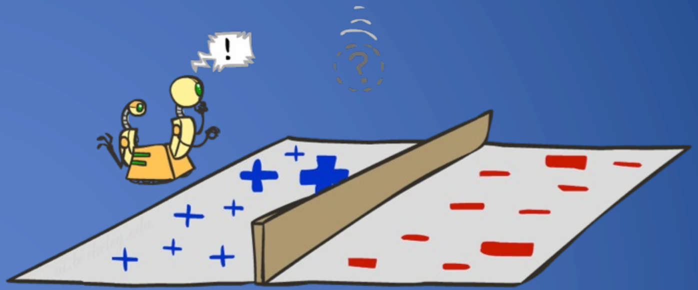
$$\boldsymbol{f(x) * w = 0}$$

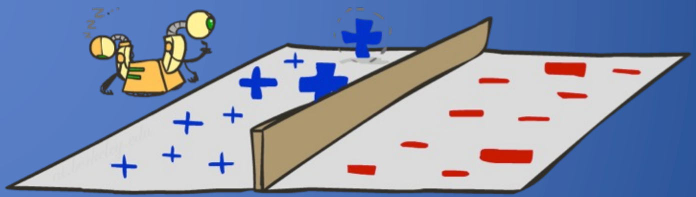
Weight Updates

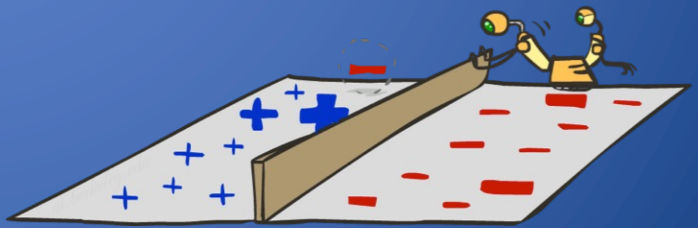


Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector







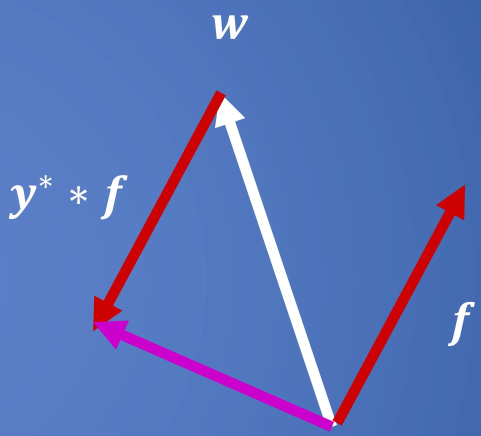
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 \; if \; w * f(x) \geq 0 \\ -1 \; if \; w * f(x) < 0 \end{cases}$$

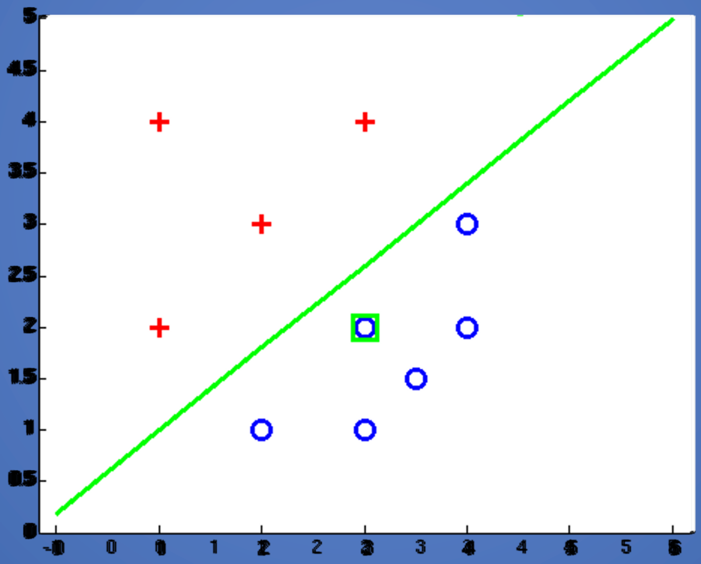
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

$$w = w + y^* * f$$



Examples: Perceptron

- Separable Case



Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class:

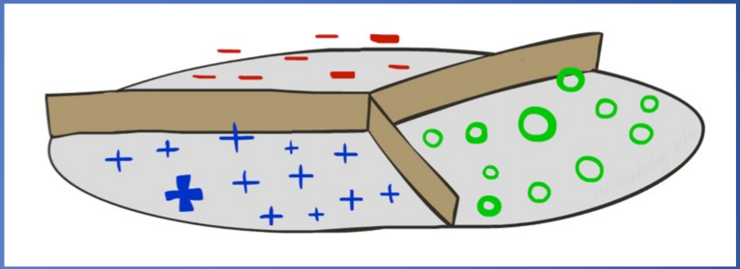
    $$w_y$$

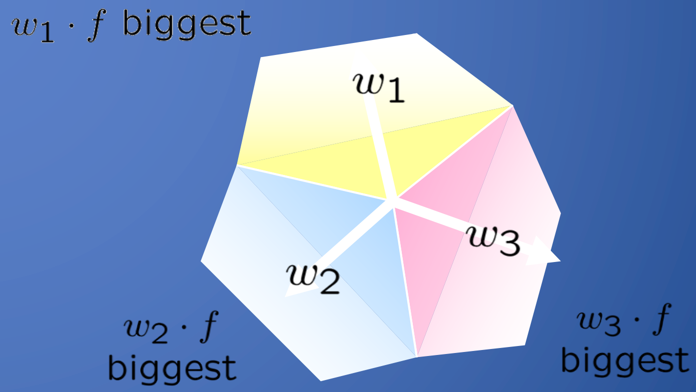
  - Score (activation) of a class y (there will be a *w* for each class):

    $$f(x) * w_y$$

  - Prediction highest score wins

    $$y = argmax_y\, f(x) * w_y$$





Binary = multiclass where the negative class has weight zero

Learning: Multiclass Perceptron

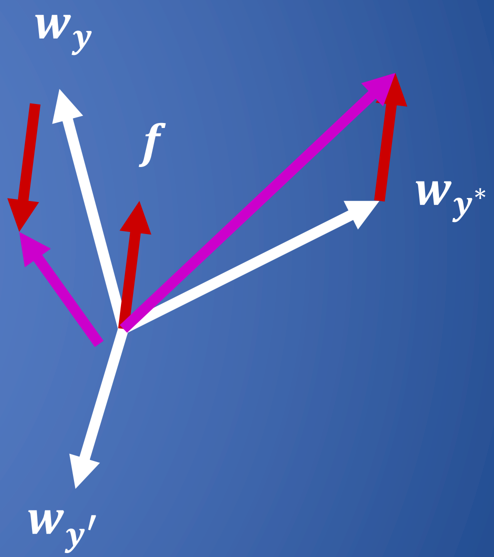
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

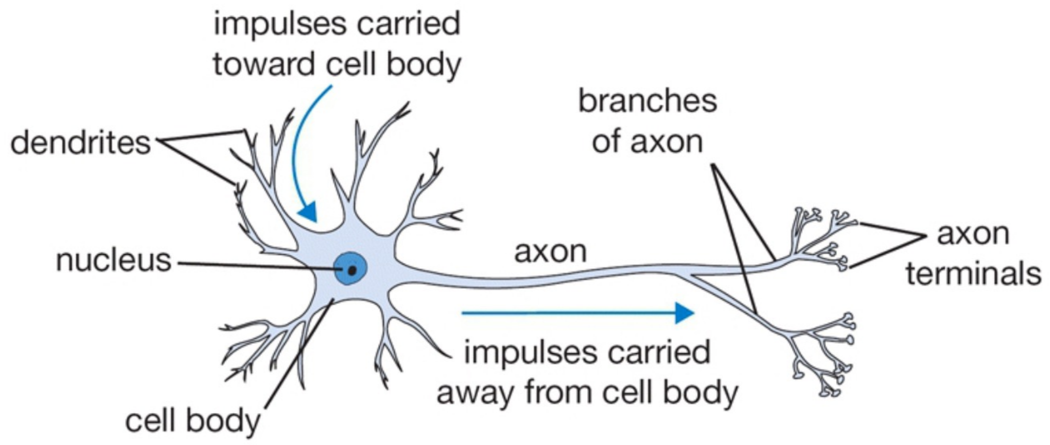
$$y = argmax_y\ f(x) * w_y$$

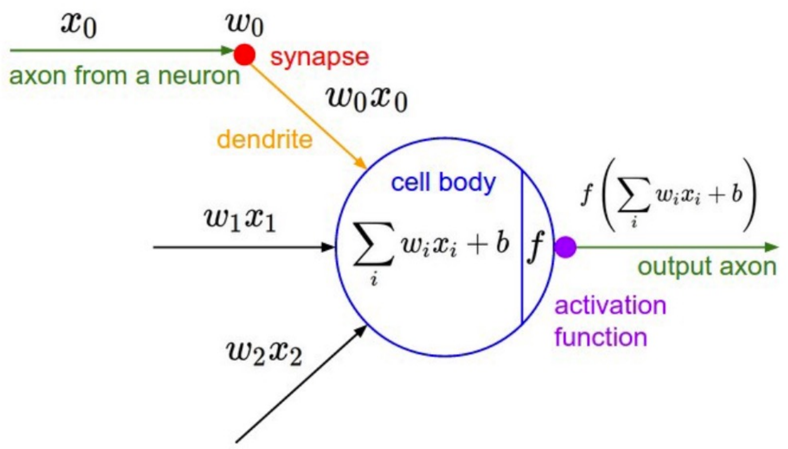
- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer:

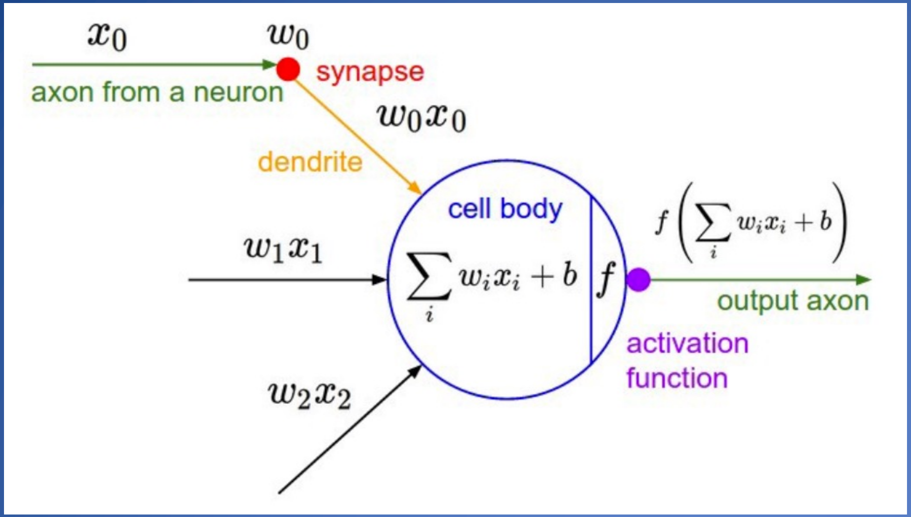
$$w_y = w_y - f(x)$$

$$w_{y*} = w_y + f(x)$$




impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body


x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$f\left(\sum_i w_i x_i + b\right)$
$w_1 x_1$
$\sum_i w_i x_i + b$
f
output axon
activation function
$w_2 x_2$

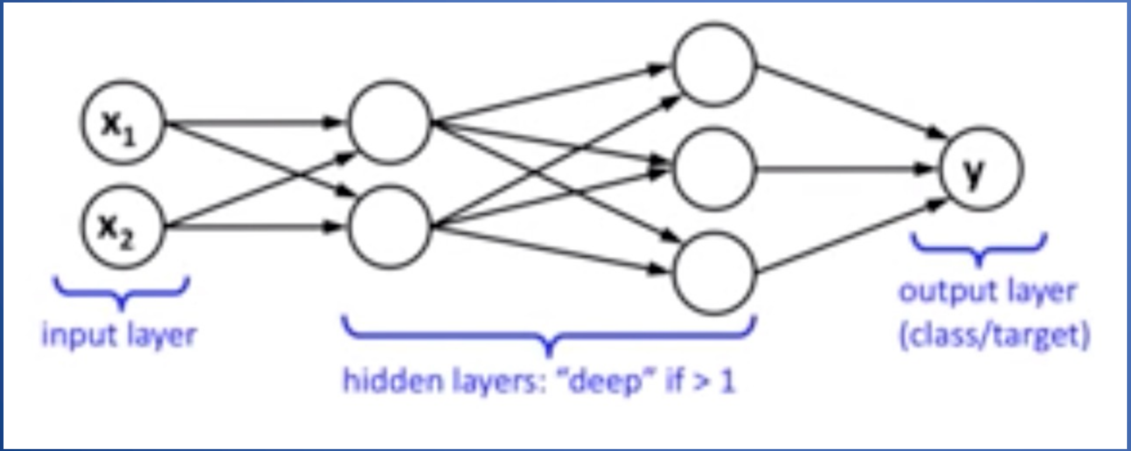


Perceptron
(linear)

Activation function
(step activation function)

$$f(s) = f\left\{\sum_i w * X\right\}$$

***f*(s) = 1,**
if *s* > 0



Activation function
(sigmoid activation function)

$$f(s) = f\left\{\sum_i w * X\right\}$$

$$f(s) = \frac{1}{1 + e^s}$$

Generalization

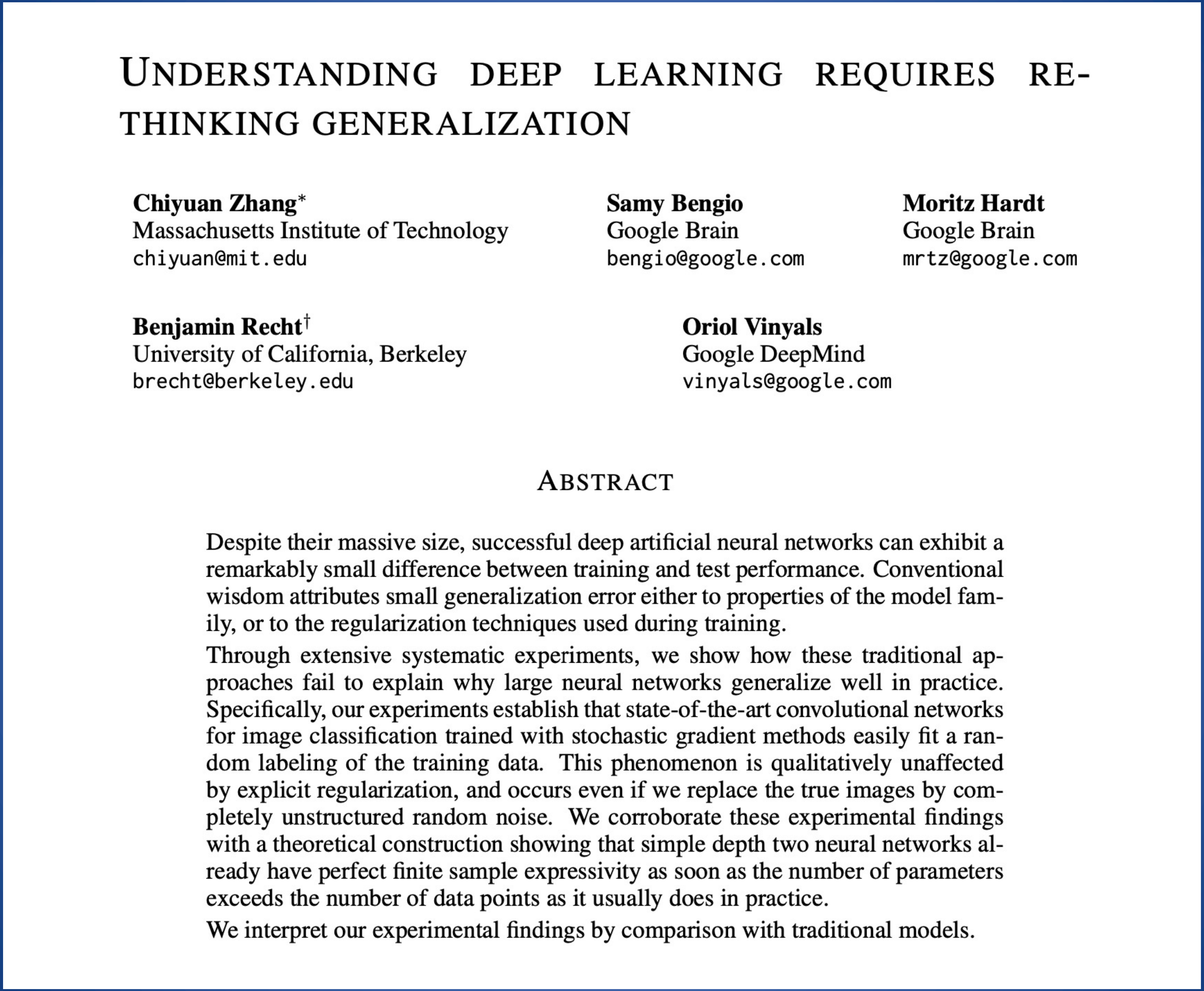
When we train a machine learning model, we do not just want it to learn to model the training data. We want it to generalize to data it has not seen before.

Generalization, therefore, is the ability of the machine learning model to perform efficiently on data it has not seen before.

If an algorithm works well on the training set but fails to generalize, we say it is overfitting. Improving generalization (or preventing overfitting) in neural nets is still a sort of dark art.

Rethinking generalization

The ICLR 2017 submission “Understanding Deep Learning required Rethinking Generalization” has certainly disrupted our understanding of Deep Learning
(see Zhang et al. 2017; Zhang et al. 2021).



UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

Chiyuan Zhang[*]
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht[†]
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a remarkably small difference between training and test performance. Conventional wisdom attributes small generalization error either to properties of the model family, or to the regularization techniques used during training.

Through extensive systematic experiments, we show how these traditional approaches fail to explain why large neural networks generalize well in practice. Specifically, our experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a random labeling of the training data. This phenomenon is qualitatively unaffected by explicit regularization, and occurs even if we replace the true images by completely unstructured random noise. We corroborate these experimental findings with a theoretical construction showing that simple depth two neural networks already have perfect finite sample expressivity as soon as the number of parameters exceeds the number of data points as it usually does in practice.

We interpret our experimental findings by comparison with traditional models.

The issue at stack

Deep artificial neural networks often have far more trainable model parameters than the number of samples they are trained on.

Yet, some of these models exhibit remarkably small generalization error. At the same time, it is certainly easy to come up with natural model architectures that generalize poorly.

→ *What is it that distinguishes neural networks that generalize well from those that do not?*

Complexity measures

Computational Learning Theory (CLT) has proposed a number of different complexity measures that are capable of controlling the generalization error:

- ✓ Structural Risk Minimization (Vapnik & Chervonenkis, 1974)
- ✓ Rademacher complexity (Bartlett & Mendelson, 2003)
- ✓ Uniform stability (Mukherjee et al., 2002; Poggio et al., 2004)

However, according to most of the scholars, when the number of parameters is large, some form of regularization is needed to ensure small generalization error.

Learners and Complexity

- There are many versions of underfit/overfit trade-off
  - Expressiveness = Representational Power
  - Representational Power = Complexity of the Learner

  Different learners have different power

- Usual trade-off:
  - More power = represent more complex systems, might overfit
  - Less power = will not overfit, but may not find “best” learner

- How can we quantify representational power?
  - Not easily…
  - One solution is VC (Vapnik-Chervonenkis) dimension

Some notation

- Let us assume our training data are i.i.d. from some distribution p(x)
- Define "risk" and "empirical risk"
  - These are just "long term" test and observed training error

$$R(\theta) = \text{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$$

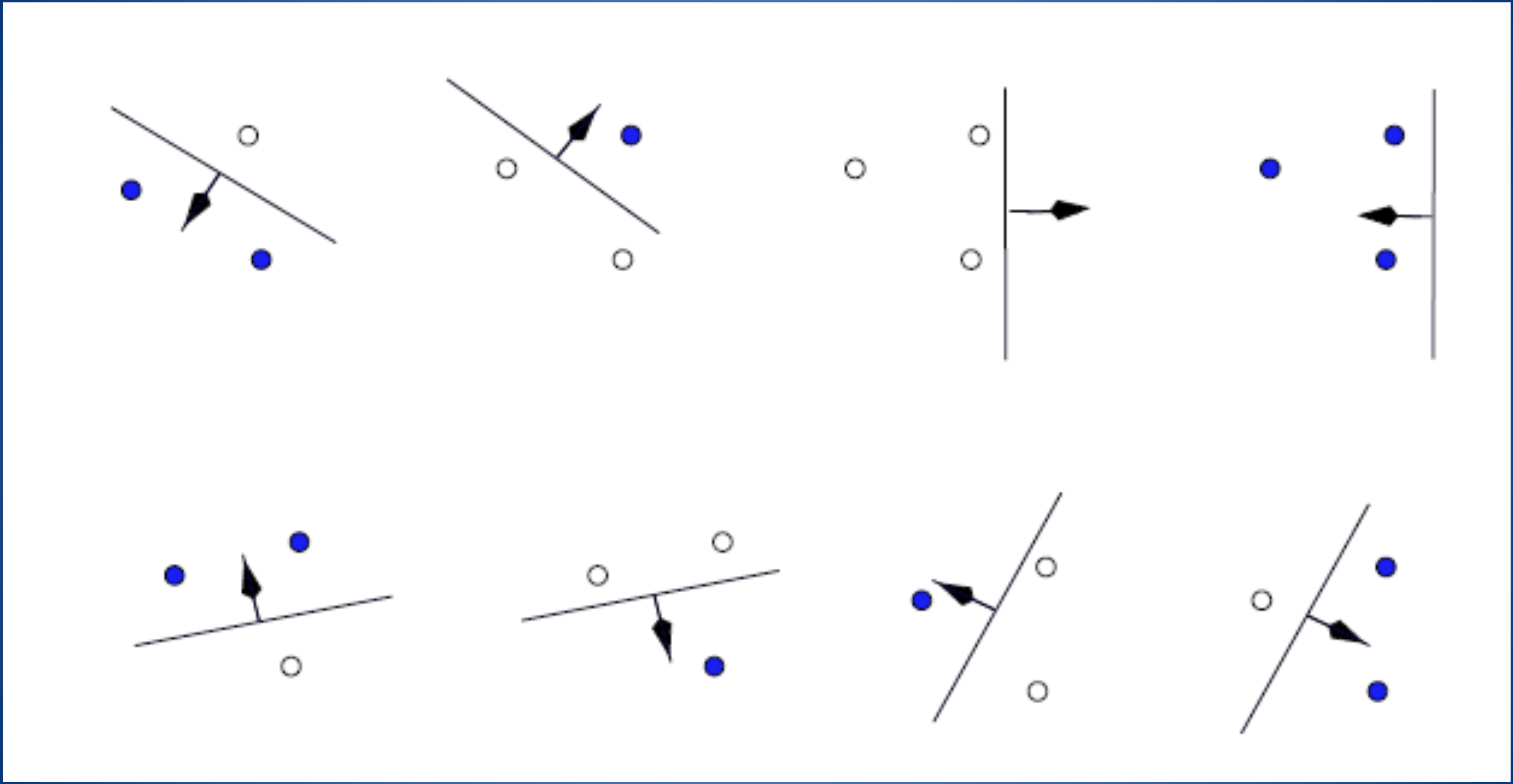
$$R^{\text{emp}}(\theta) = \text{TrainError} = \frac{1}{m}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$$

- How are these related? Depends on overfitting…
  - Underfitting regime: pretty similar…
  - Overfitting regime: test error might be lots worse!

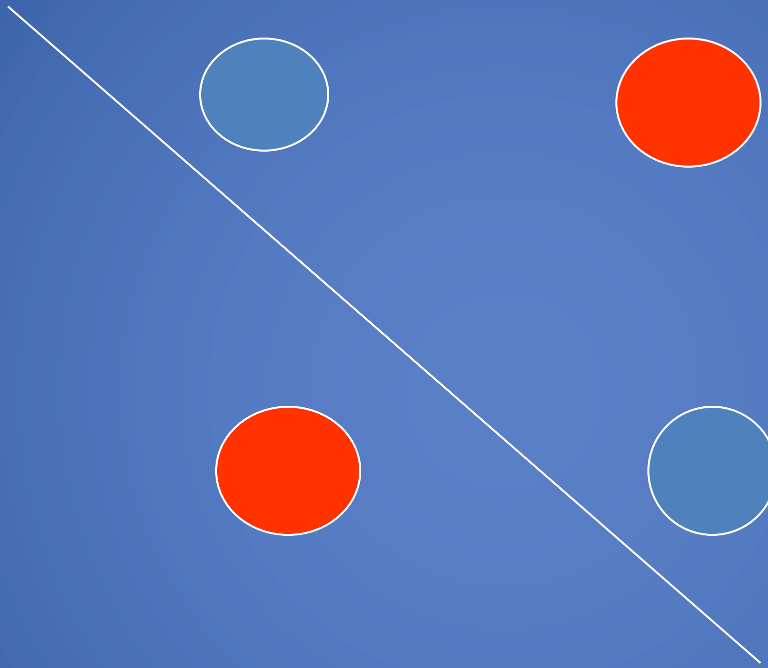
VC Dimension

- VC dimension for a set of functions {*f*(a)} is defined as the maximum number of training points that can be shattered by {*f*(a)}.
- If the VC dimension is *h*, then there exists at least one set of *H* points that can be shattered. But not necessary for every set of *H* points.

A linear function has VC dimension 3



8 possible labeling of 3 points can be separated by lines



Yet a linear function cannot separate the labeling of these four points using a line. Thus the VC dimension of a line is 3.

VC Dimension and Structural Risk Minimization

- Given some machine **f**, let *H* be its VC dimension.
- *H* is a measure of **f**'s power (*H* does not depend on the choice of training set)
- Vapnik showed that with "high probability" 1-η represents the "representational power" of classifier

$$R(\theta) = \text{TestError} = \mathbb{E}[\delta(c \neq \hat{c}(x\,;\,\theta))]$$

$$R^{\text{emp}}(\theta) = \text{TrainError} = \frac{1}{m}\sum_i \delta(c^{(i)} \neq \hat{c}(x^{(i)}\,;\,\theta))$$

- **Structural Risk Minimization**

$$\text{TestError} \leq \text{TrainError} + \sqrt{\frac{H\log(2m/H) + H - \log(\eta/4)}{m}}$$

What does the VC dimension measure?

- Is it the number of parameters?

  Related but not really the same.

- I can create a machine with one numeric parameter that really encodes 7 parameters
- And I can create a machine with 7 parameters which has a VC-dim of 1
- *Filippos private opinion: it often is the number of parameters that counts.*

Using VC-dimensionality

People have worked hard to find VC-dimension for...

- Decision Trees
- Perceptrons
- Neural Nets
- Support Vector Machines
- And many many more

All with the goals of

1. Understanding which learning machines are more or less powerful under which circumstances
2. Using Structural Risk Minimization to choose the best learning machine

Regularization

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error, possibly at the expense of increased training error。

The two variables controlling regularization

- ✓ Network model
- ✓ Learning method

Regularization

- ✓ *Regularization by construction* is present both in the training and inference stages
- ✓ *Regularization by training* is not present in the inference path

Regularization by training

✓ *Explicit regularization*:

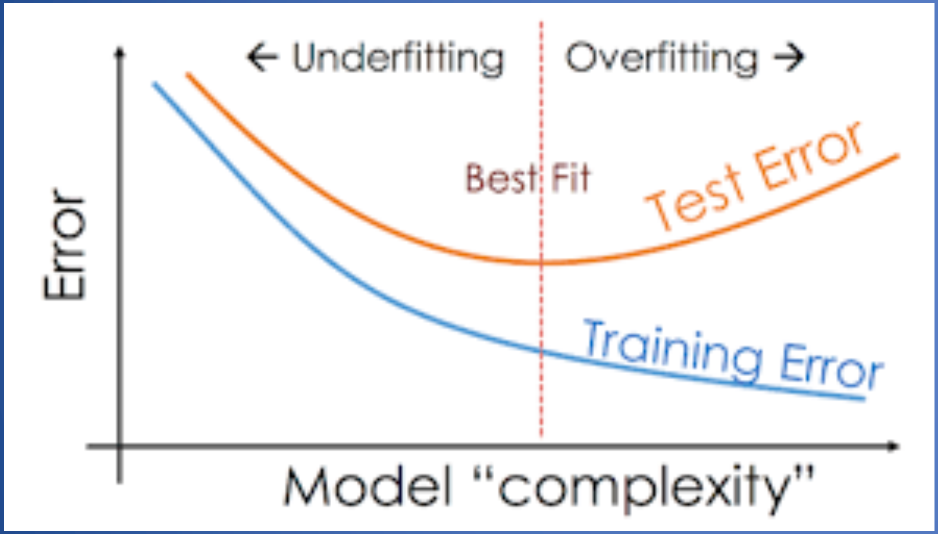
- early stopping
- dropout
- weight decay/weight sharing
- data augmentation

→ In all these cases the goal of regularization is to improve generalization.

✓ *Implicit regularization*:

- SGD
- batch normalization

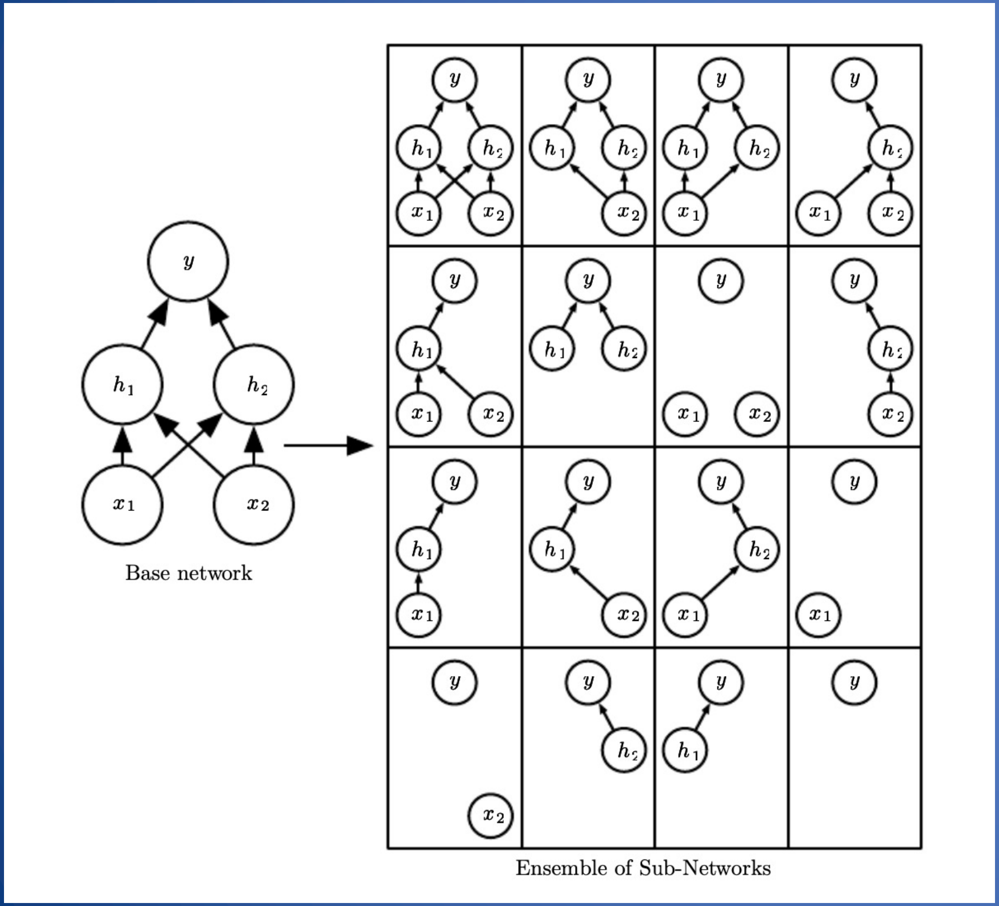
Early stopping



It is an hyperparameter controlling the effective capacity of the model by determining how
many steps it can take to fit the training set.

The additional cost to early stopping is the need to maintain a copy of the best parameters.

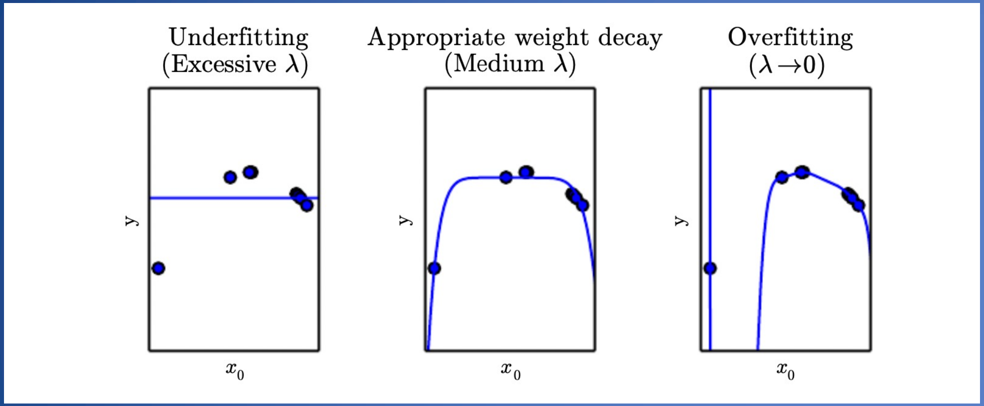
Dropout



Dropout trains the ensemble of networks consisting of all sub-networks that can be formed by removing non-output units from an underlying base network.

Dropout trains an ensemble of models that share hidden units. This means each hidden unit must be able to perform well regardless of which other hidden units are in the model.

Weight decay/sharing



$$J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w},$$

We minimize a sum comprising both the mean squared error on the training and a criterion J(w) that expresses a preference for the weights to have smaller squared L2 norm.

Weight decay penalizes model parameters for deviating from the fixed value of zero.

Weight sharing forces sets of parameters to be equal.

Data augmentation

The main task facing a classifier is to be invariant to a wide variety of transformations. We can generate new (x, y) pairs easily just by transforming the x inputs in our training set.

Operations like translating the training images a few pixels in each direction, rotating the image or scaling the image can often greatly improve generalization, even if the model has already been designed to be partially translation invariant by using the convolution and pooling techniques.

Injecting noise in the input to a neural network can also be seen as a form of data augmentation.

Discovery (1)

1) Deep neural networks, when trained on a completely random labeling of the data, can easily achieve 0 training error.

2) The effective capacity of neural networks is large enough for a brute-force memorization of the entire training set.

3) The test error will be of course no better than *random chance* insofar as there is no correlation between the training labels and the test labels.

4) Even optimization on random labels remains easy: training time increases only by a small constant factor compared with training on the true labels.

5) Randomizing labels is solely a data transformation, leaving all other properties of the learning problem unchanged.

Discovery (2)

If the model architecture itself is not a sufficient regularizer, it remains to see how much explicit regularization, such as dropout or data augmentation, helps.

→ "Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error" (Zhang and al. 2017)

Four different notions of generalization

Definition 1: Error Response to Validation and Real Data

We can define it as the behavior of our system in response to validation data, i.e. data that we have not included as part of the training set.

We might be more ambitious and define it as behavior when the system is deployed to analyze real world data: we essentially would like to see our trained system perform accurately in the context of data it has never seen.

Four different notions of generalization

Definition 2: Model sparsity

A second definition is based on the idea of Occam’s Razor, i.e. the simplest among the hypotheses is the best one.

Feature selection simplifies a machine learning problem by choosing which subset of the available features should be used.

Four different notions of generalization

Definition 3: Fidelity in Generating Models

A third definition is based on the system's ability to recreate or reconstruct the features.

This is the approach taken by generative models. If a neural network is able to accurately generate realistic images, then it is able to capture the concept of that image in its entirety.

Four different notions of generalization

Definition 4: Effectiveness in Ignoring Nuisance Features

This definition involves the notion of ignoring nuisance variables, i.e. a system is able to generalize whenever it is able to ignore nuisance features for its tasks.

Remove away as many features as possible until you cannot remove any more.

SGD & Generalization

Methods that do not seem to have anything to do with generalization issues such as Stochastic Gradient Descent in fact do contribute

A number of works have used a variety of techniques to conclude that the loss functions of deep network have no bad (or a few) local minima (even though may instead have many saddle points)

SGD acts as an implicit regularizer.: for linear models, SGD always converges to a solution with a small norm.

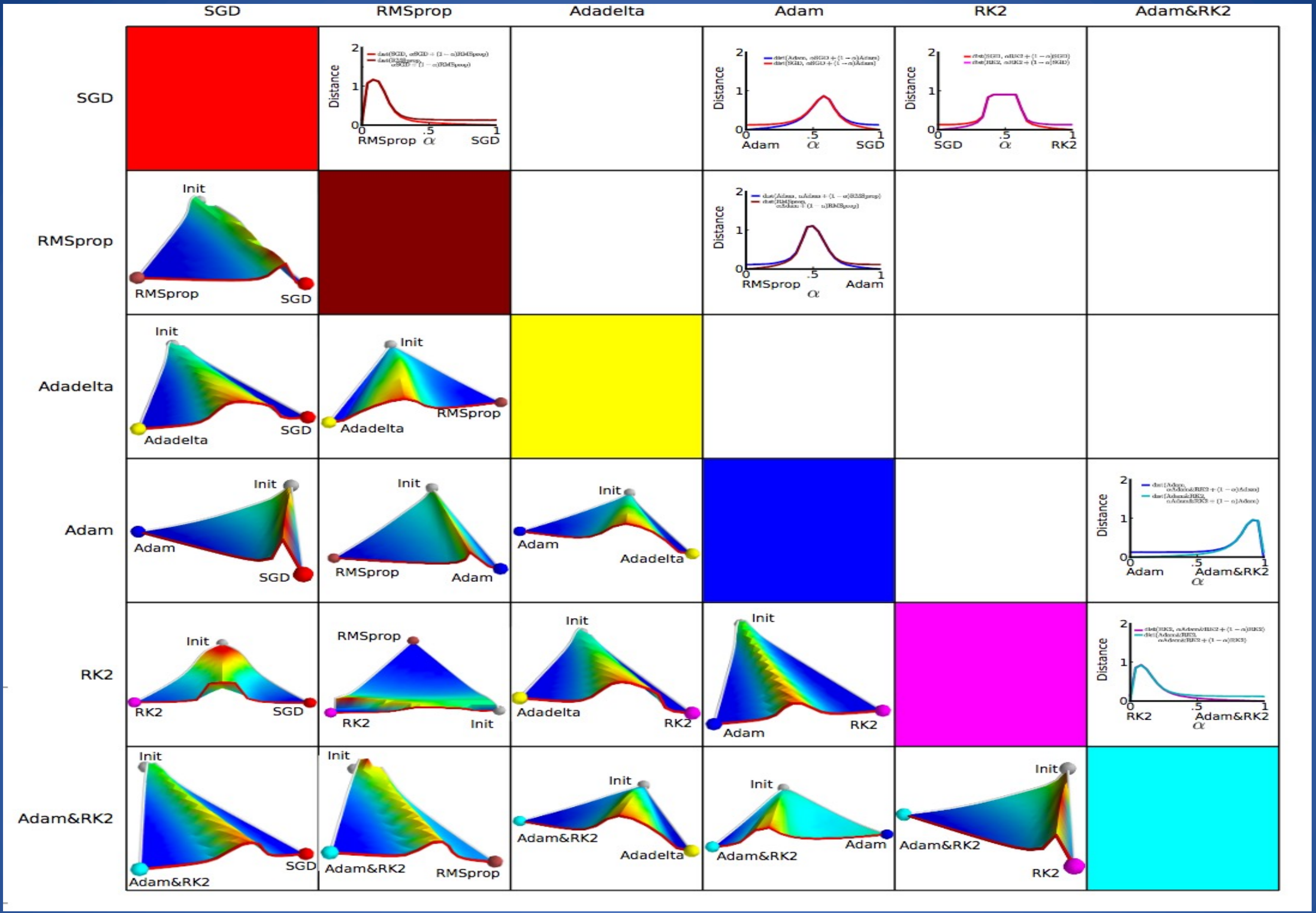
Deep Learning loss surfaces

The success of deep learning critically depends on how well we can minimize loss functions: understanding the geometry of these loss functions and how optimization algorithms traverse them is thus of vital importance.

1) Different optimization algorithms (SGD, SGDM, RMSProp, Adadelta, Adam) find different local minima

2) The loss surfaces of the same algorithm from different initializations are remarkably characteristic of the optimization algorithm

3) Batch normalization is key to obtaining this consistency in the projected loss surface



To investigate the shape of the loss function, it is possible to compute the value of the loss function for weight vectors interpolated between the initial weights, the final weights for one algorithm, and the final weights for a second algorithm for each pairing of algorithms for the VGG (Visual Geometry Group) network.

Deep Learning loss surfaces

Different optimization algorithms find different types of local minima

a. It is possible to identify some stereotypical features for the final points found by different algorithms

b. However, the generalization accuracy of these different final points on validation data was not different between algorithms

c. Moreover it does not appear to be any relationship between the weight initialization and the validation accuracy

How to understand the notion of effective capacity

1. The effective capacity of successful neural network architectures is large enough to shatter the training data

2. Consequently, these models are in principle rich enough to memorize the training data

3. Therefore, traditional measures of model complexity struggle to explain the generalization ability of large artificial neural networks

4. Yet, optimization continues to be empirically easy even if the resulting model does not generalize

Can we thrust AI?

Neural Networks are black boxes to the extent that:

- ✓ why and how generalization and, therefore, learning is achieved is not clear (see, for instance, Zhang 2017; 2021);
- ✓ the behavior of NN cannot be easily interpreted to the extent that we cannot read their operation step-by-step by using a reverse engineering approach (e.g. what is the contribution of the example n. 13412 to the final output?);
- ✓ NN are not able to justify their decisions by providing plausible reasons or explanations for their own decisions: therefore, NN cannot enter into a realm of rhetorical negotiation between the human and the artificial actors (please make a comparison with decision trees).

Thanks for your attention!